[illegible]

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD-A203 713

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Valiant's Maximum Algorithm with Sequential Memory Accesses		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Robert Cypher		8. CONTRACT OR GRANT NUMBER(s) N00014-86-K-0264
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Washington Department of Computer Science Seattle, Washington 98195		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Information Systems Program Arlington, VA 22217		12. REPORT DATE April 1988
		13. NUMBER OF PAGES 2
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this report is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

parallel algorithms, maximum finding, analysis of algorithms

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper examines the performance of Valiant's PRAM algorithm for finding the maximum of a set of numbers, assuming that a modified PRAM model is used. The modified PRAM is like a standard CRCW PRAM, except that multiple read or write requests to a single memory location are handled sequentially. It is shown that using this model, Valiant's algorithm requires O(sqrt(N)) time to find the maximum of N numbers using N processors, and that it does require time proportional to sqrt(N) infinitely often. [illegible]

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Valiant's Maximum Algorithm with Sequential Memory Accesses

Robert Cypher
Department of Computer Science, FR-35
University of Washington
Seattle, Washington 98195

TR 88-03-08
April 1988

Abstract

· This paper examines the performance of Valiant's PRAM algorithm for finding the maximum of a set of numbers, assuming that a modified PRAM model is used. The modified PRAM is like a standard CRCW PRAM, except that multiple read or write requests to a single memory location are handled sequentially. It is shown that using this model, Valiant's algorithm requires O(sqrt(N)) time to find the maximum of N numbers using N processors, and that it does require time proportional to sqrt(N) infinitely often.

This research funded in part by the Office of Naval Research Contract N00014-86-K-0264, National Science Foundation Grant CCR-8416878, and the Air Force Office of Scientific Research Contract 88-0023.

Valiant's Maximum Algorithm with Sequential Memory Accesses

Valiant has created a PRAM algorithm for finding the maximum of N numbers that uses P = N processors and operates in O(loglog P) time [1]. That algorithm consists of O(loglog P) stages, each of which takes constant time. At the start of each stage, a certain number of candidates for the maximum are present. These candidates are divided into roughly equal-sized groups, and the maximum item from each group is calculated. These group maxima form the set of candidates for the following stage. Within each group, all possible pairs of items are compared, with one processor being responsible for each such pair. As a result, a group with X items requires that $\binom{X}{2}$ processors be assigned to it. The group sizes are chosen to be as large as possible given the limited number of processors that are available.

Valiant's algorithm requires that in unit time a large number of processors can read from or write to a single memory location. This paper examines the performance of Valiant's algorithm when this assumption is changed. In particular, it will be assumed that multiple requests to a single memory location are handled sequentially. This analysis arises as a result of investigations by L. Snyder [2].

Let N(t) be the number of items left at time (stage) t, and let R(t) be the number of groups that the N(t) items are divided into, and let S(t) be the size of the largest of these groups. For instance, assuming P processors and N = P numbers to begin with, N(0) = P, $R(0) = \lceil P/3 \rceil$, and S(0) = 3. Because each group contibutes one candidate to the next stage, R(t) = N(t+1) for all values of t. Also, $S(t) = \lceil N(t)/R(t) \rceil$ for all values of t.

In the original analysis of Valiant's algorithm, each stage required constant time. With the current assumptions about sequential access to memory locations, stage t requires O(S(t)) time to complete. Thus, the time for the entire algorithm depends on the sum of the O(loglog P) values of S(t). It will be shown that if P processors are available, and N = P numbers are examined for the maximum, then the algorithm operates in $O(P^{1/2})$ time. In fact, the sum of the S(t)'s is never more than $(2P)^{1/2} + o(P^{1/2})$ and it is greater than $(2P)^{1/2}$ infinitely often.

To see that the sum of the S(t)'s for all O(loglog P) values of t is never more than $2P^{1/2} + o(P^{1/2})$, the following theorem is needed:

Theorem: For all values of t, $S(t) \leq (2P/R(t))^{1/2} + 2$.

Proof: Because S(t) is the size of the largest group at time t, the smallest group at time t has at least S(t) - 1 items and thus each group needs at least $\binom{S(t)-1}{2}$ processors. Because there are R(t) groups, $\binom{S(t)-1}{2} \leq P/R(t)$. From this, it follows that $(S(t) - 1)(S(t) - 2) \leq 2P/R(t)$ and that $S(t) \leq (2P/R(t))^{1/2} + 2$.

Note that as a corollary to the theorem, $S(t) \leq (2P)^{1/2} + 2$ for all values of t because $R(t) \geq 1$. Now consider the values of S(t) for all O(loglog P) values of t. Either all of the values of S(t) are $\leq P^{1/3}$, in which case the sum of the values is of $O(P^{1/3}$ loglog P), or there is some largest value of t s.t. $S(t) > P^{1/3}$. Let this value of t be denoted u. Thus, $S(u) > P^{1/3}$ and $S(t) \leq P^{1/3}$ for $t > u$. The sum of the S(t)'s for all values of $t > u$ is of $O(P^{1/3}$ loglog P). From the corollary, $S(u) \leq (2P)^{1/2} + 2$. Next, because $S(u) > P^{1/3}$, $R(u\text{-}1) = N(u) > P^{1/3}$, and from the theorem, $S(u\text{-}1) \leq (2P/R(u\text{-}1))^{1/2} + 2 \leq (2P^{2/3})^{1/2} + 2 = O(P^{1/3})$. Finally, because $S(t\text{-}1) \leq S(t)$ for $t < u$ (this is easy to show), the sum of the S(t)'s for $0 \leq t < u$ is of $O(P^{1/3}$ loglog P). As a result, the sum of the S(t)'s for all values of t is $\leq (2P)^{1/2} + o(P^{1/2})$.

To see that the sum of the S(t)'s is greater than $(2P)^{1/2}$ infinitely often, consider the special case when at every stage, all of the groups have the same size. In this case, $S(0) = 3$, $S(1) = 7$, $S(2) = 43$, $S(3) = 1807$, etc.. In general, $S(t) = 1 + 2\prod_{x=0}^{t-1} S(x)$. Consider the values of P given by the sequence $P_0 = 1$ and $P_{j+1} = P_j(2P_j + 1)$. For example, $P_0 = 1$, $P_1 = 3$, $P_2 = 21$, $P_3 = 903$ and $P_4 = 1{,}631{,}721$. It can be shown that for any of these P_j's, the final stage will have N(t) items where $N(t)(N(t) - 1)/2 = P_j$, so for this value of t, $R(t) = 1$ and all of the processors are used to find the maximum of the N(t) remaining items. However, the N(t) that satisfies the above equality is such that $N(t) > (2P_j)^{1/2}$.

References

[1] L. G. Valiant, Parallelism in Comparison Problems. *Siam Journal of Computing* 4(3):348-355, 1975.

[2] L. Snyder, Type Architectures, Shared Memory, and the Corollary of Modest Potential. *Annual Review of Computer Science* 1:289-317, 1986.